第 1 章习题

1.1 选择一个单位的工资表，指出其中的元素、元素的字段以及元素之间的关系，并给出一些最基本的运算。

1.2 描述数据结构、逻辑结构、存储结构和运算的有关概念及其相互之间的关系。

1.3 已知一个群体中有 n 个人，这些人之间可能存在同学关系，请用一个数据模型来描述这一关系，并给出可能基本运算。

1.4 描述算法所具备的基本特征，并指出算法与程序之间的差异。

1.5 计算序列各程序段的时间复杂度。

（1）
```
for    ( i=0; i<n; i++ )
          for    ( j=i; j<n; j++) x++;
```
（2）
```
i=n;
    while ( i>1 ) i/= i/2;
```
（3）
```
for ( i=1; i<=n; i++)
          for ( j=1; j<=n; j++)
               for ( k=1; k<=n; k++)
                    x++;
```
（4）
```
for ( i=1; i<n; i++ )
           for ( j=1; j<n; j++ ) x++;
     for ( k=1; k<n; k++ ) x++;
```
（5）
```
for ( i=1; i<n; i++ )
        {    j=i;
             while ( j<n )    j*=2;
        }
```

第 2 章习题

2.1 若将顺序表中记录其长度的分量 listlen 改为指向最后一个元素的位置 last，在实现各基本运算时需要做那些修改？

2.2 试用顺序表表示较多位数的大整数，以便于这类数据的存储。请选择合适的存放次序，并分别写出这类大数的比较、加、减、乘、除等运算，并分析算法的时间性能。

2.3 试用顺序表表示集合，并确定合适的约定，在此基础上编写算法以实现集合的交、并、差等运算，并分析各算法的时间性能。

2.4 假设顺序表 L 中的元素递增有序，设计算法在顺序表中插入元素 x，要求插入后仍保持其递增有序特性，并要求时间尽可能少。

2.5 假设顺序表 L 中的元素递增有序，设计算法在顺序表中插入元素 x，并要求在插入后也没有相同的元素，即若表中存在相同的元素，则不执行插入操作。

2.6 设计算法以删除顺序表中重复的元素，并分析算法的时间性能。

2.7 假设顺序表 L 中的元素按从小到大的次序排列，设计算法以删除表中重复的元素，并要求时间尽可能少。要求：

（1）对顺序表（1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,8,9）模拟执行本算法，并统计移动元素的次数。

（2）分析算法的时间性能。

2.8 若递增有序顺序表 A、B 分别表示一个集合，设计算法求解 A=A∩B，并分析其时间性能。

2.9 递增有序顺序表 A、B 分别表示一个集合，设计算法求解 A=A－B，并分析其时间性能。

2.10 假设带头结点的单链表是递增有序的，设计算法在其中插入一个值为 x 的结点，并保持其递增特性。

2.11 设计算法以删除链表中值为 x 的元素结点。

2.12 设计算法将两个带头结点的单循环链表 A,B 首尾相接为一个单循环链表 A。

2.13 假设链表 A、B 分别表示一个集合，试设计算法以判断集合 A 是否是集合 B 的子集，若是，则返回 1，否则返回 0，并分析算法的时间复杂度。

2.14 假设递增有序的带头结点的链表 A、B 分别表示一个集合，试设计算法以判断集合 A 是否是集合 B 的子集，若是，则返回 1，否则返回 0，并分析算法的时间复杂度。

2.15 假设链表 A、B 分别表示一个集合，设计算法以求解 C= A∩B，并分析算法的时间复杂度。

2.16 假设递增有序的带头结点的链表 A、B 分别表示一个集合，设计算法以求解 C= A∩B，并分析算法的时间复杂度。

2.17 假设递增有序的带头结点的链表 A、B 分别表示一个集合，设计算法以求解 A= A∩B，并分析算法的时间复杂度。

2.18 假设递增有序的带头结点的单循环链表 A、B 分别表示两个集合，设计算法以求解 A= A∪B，并分析算法的时间复杂度。

2.19 假设链表 A、B 分别表示两个集合，设计算法以求解 C= A∪B，并分析算法的时间复杂度。

2.20 设计算法将两个递增有序的带头结点的单链表 A、B 合并为一个递增有序的带头结点的单链表，并要求算法的时间复杂度为两个表长之和的数量级。

2.21 设计算法将链表 L 就地逆置，即利用原表各结点的空间实现逆置。

2.22 设计算法将两个递增有序的带头结点的单链表 A、B 合并为一个递减有序的带头结点的单链表，并要求算法的时间复杂度为两个表长之和的数量级。

2.23 设计算法以判断带头结点的双循环链表 L 是否是对称的，即从前往后和从后往前的输出序列是相同的。若对称，返回 1，否则返回 0。

2.24 设计算法将带头结点的双循环链表 L 就地逆置，即利用原表各结点的空间实现逆置。

第 3 章习题

3.1 对一个栈的输入序列 $a_1, a_2, a_3, \ldots, a_n$，，称由此栈依次出栈后所得到的元素序列为栈的合法输出序列。例如，假设栈 S 的一个输入序列为 1,2,3,4,5，则可得到多个输出序列，例如，1,2,3,4,5 就是一个合法的输出序列，同理，5,4,3,2,1 和 3,2,1,4,5 也分别是其合法的输出序列。分别求解下列问题：

（1）判断序列 1,3,4,5,2 是否是合法的输出序列。

（2）对输入序列 1,2,3,4,5，求出其所有的合法的输出序列。

（3*）设计算法以判断对输入序列 1,2,3,… ,n，序列 $a_1, a_2, a_3, \ldots, a_n$ 是否是该栈的合法的输出序列（假设输出序列在数组 A 中）。

3.2 如果顺序栈中的第二个分量是记录元素个数的变量而不是栈顶指针，应如何实现各算法？

3.3 设计出链栈的各基本问题求解的算法，并分析其时间复杂度。

3.4 对一个合法的数学表达式来说，其中的各大小括号“{”，“}”，“[”，“]”，“(”和“)”应是相互匹配的。设计算法对以字符串形式读入的表达式 S，判断其中的各括号是否是匹配的。

3.5 对表达式 5+3*(12+4)/4-8，依次画出在求解过程中的各步骤中的栈的状态。

3.6*设计算法以求解所读入的表达式的值。假设数据类型为整型，并且仅包含加减乘除四则运算。

3.7*设计算法以求解所读入的表达式的值。假设数据类型为浮点型，并且仅包含加减乘除四则运算。

3.8 用一个数组、头指针和元素个数合在一起所构成的结构来存储顺序队列，设计算法以实现队列的各运算。

3.9 对教材中所讨论的循环队列及其约定，给出求解队列中元素个数的表达式。

3.10 如果对循环队列采用设置运算标志的方法来区分队列的满和空的状态，试给出对应的各运算的实现。

3.11 如果采用带尾指针的单循环链表作为队列的存储结构，设计算法以实现队列的各运算。

3.12 写出下面程序或调用的结果:

```
(1)  void  P1(int W);
          {  int A,B;
             A=W-1;  B=W+1;
             cout<<A<<B;
          }
     void  P2(W int );
```

```
    {  int A,B;
       A=2*W;  B=W*W;
       P1(A);  P1(B);
       cout<<A<<B;
    }
  调用  P2(5);
(2)  int  Hcf(int M,int N)
    { int H;
       while (N!=0)
          {
             H=M % N;
             M=N; N=H
          } ;
       cout<<M;  return M;
    } ;
  调用  cout<<Hcf(100, 350);
       cout<<Hcf(200, 49);
(3)  int Hcf(int M, int N)
    {   int H;
       while (N!=0)
          {
             H=M mod N;
             M=N; N=H
          }
       return M;
    }
  void  reduce(int M1, int N1; int *M2, int *N2)
    {  int R;
       R=Hcf(M1,N1);
        *M2=M1 / R;  *N2=N1/ R;
       cout<<M1<<'/'<<N1<<'='<<M2<<'/'<<N2;
    }
 调用 reduce(100,200, X,Y);
     reduce(300,550, M,N);
```

3.13 阅读下列程序，并写出其运行结果:

```
(1)  void  P(int W)
    {
      if (W>0)
        {   P(W-1);
            cout<<W;
        }
```

```
        }
    调用 P(4);
(2)   void  P(int W)
        {
            if (W>0)
              {   cout<<W;
                  P(W-1);
              }
        }
      调用 P(4);
(3)   void  P(int W)
        {
          if (W>0)
            {  cout<<W;
               P(W-1);
               cout<<W;
            }
        }
    调用 P(4);
(4)   void  P(int W)
        {
            if (W>0)
              {   P(W-1);
                  P(W-1);
                  cout<<W;
               }
        }
    调用 P(4);
(5)   int   F(int N)
        {
            if ( N==0) F=0;
            else  if (N==1)
                   return 1;
                  else  return F(N-1)+2*F(N-2);
        }
    调用 cout<<F(5);
(6)   void  P(int N,int *F);
        {
            if (N==0)
             *F=0;
            else  {  P(N-1, *F); F=F+N;  }
```

```
    }
```
调用 P(4,*M); cout<<*M

3.14 求解下面各调用的结果，并指出算法的功能。

(1)
```
void  PrintRV(int N)
    {
        if (N>0)
          {  cout<<N % 10;
             PrintRV(N / 10);
          }
    } ;
```
调用 PrintRV(12345);

(2)
```
void  PC(int M, int N; int *K );
    {
        if (N==0)
          *K=1;
        else {  PC(M-1,N-1,*K); *K=*K*M / N;  }
    } ;
```
调用 PC(10,4,*M); cout<<*M;

(3)
```
int   SS(int N)
    {
        if (N==0 ) return 100;
       else  return SS(N-1)+N*N;
    }
```
调用 cout<<SS(5);

(4)
```
int   ACM(int M, int N)
    {
      if (M==0) return N+1;
      else  if (N==0) return ACM(M-1,1);
            else return ACM(M-1,ACM(M,N-1));
    }
```
调用 cout<<ACM(2,2);

3.15 对下面的函数定义，证明下面有关程序功能描述的正确性。

(1)
```
void  P(int N)
    {
    if (N>0)
      {  P(N-1);
         cout<<N;
         P(N-1);
      }
    }
```
a.在调用 P(N)所产生的输出序列中，当且仅当其序号为奇数时，输出项为 1。

b.在调用 P(N)所产生的输出序列中，输出 2 的项数为 2^{n-2}。

(2)下面函数的设计目标是求整型数组 A 中下标从 i 到 j 的各元素的最大值，请判断该函数的正确性。另外，函数中所定义的局部变量是否是必需的？

```
int Max(int i, int j)
    { int M1,M2,Mid;
        if (i==j ) return A[I];
        else { Mid=(i+j) / 2;
               M1=Max(i,Mid);
               M2=Max(Mid+1,j);
               if (M1>M2 )return M1;
                     else return M2;
             }
    }
```

3.16 已知 ACKMANN 函数定义如下，试分别编写出求解该函数的递归过程和递归函数。

$$\begin{cases} Ack(m,n)=n+1 & m=0 \\ Ack(m,n)=Ack(m-1,1) & n=0 \\ Ack(m,n)=Ack(m-1,Ack(m,n-1)) & m>0,n>0 \end{cases}$$

3.17 将下面递归程序转换为等价的非递归程序。

```
(1) void  P(int N)
    {
      if (N>0)
        {  cout<<N;
           P(N-1);
           cout<<N;
        }
    }
(2) void  P(int N)
    {
      if (N>0)
       {  P(N-1);
          cout<<N;
          P(N-1);
       }
    }
(3) int  f(int N)
   {
      if (N==0) return 0;
      else if (N==1) return 1;
      else return f(1)+f(2);
    }
```

第 4 章习题

4.1 设计算法以判断链串 S1 是否是链串 S2 的子串，若是子串，返回 1，否则返回 0。

4.2 设计算法以比较链串 S1 和链串 S2 的大小，若 S1<S2，返回-1；若 S1＝S2，返回 0；否则返回 1。

4.3 设计算法以判断两个顺序串是否相等，若相等，返回 1，否则返回 0。

4.4 已知数组 A[n,n]是对称的，完成下列任务：

① 设计算法将 A[n,n]中的下三角中的各元素按行优先次序存储到一维数组 B 中。

② 对任意输入的 A 数组中的元素的下标 i,j，求解出该元素在 B 中的存储位置。

4.5 已知数组 A[n,n]的上三角部分的各元素均为同一个值 v0，，完成下列任务：

① 设计算法将 A[n,n]中的下三角中的各元素按行优先次序存储到一维数组 B 中，并将 v0 存放到其后面。

② 对任意输入的 A 数组中的元素的下标 i,j，求解出该元素在 B 中的存储值。

4.6 对两个以三元组形式存储的同阶稀疏矩阵 A,B，设计算法求 C=A+B。

4.7 已知广义表 L=(a,(b,c,d),c),运用 head 和 tail 运算组合，取出原子 d 的运算是什么？

4.8 广义表(a,(a,b),d,e,((i,j),k))的长度是多少？

4.9 广义表与线性表的主要区别是什么？

4.10 求下列广义表操作的结果：

① tail[head[((a,b),(c,d))]];

② tail[head[tail[((a,b),(c,d))]]]

4.11 利用广义表的 head 和 tail 操作写出如上题的函数表达式,把原子 banana 分别从下列广义表中分离出来.

① L =(((apple))),((pear)),(banana),orange);

② L =(apple,(pear,(banana),orange));

第 5 章习题

5.1 画出由 4 个结点所构成的所有形态的树（假设是无序树）。

5.2 已知一棵树的度为 4，其中度为 4 的结点的数目为 3，度为 3 的结点的数目为 4，度为 2 的结点的数目为 5，度为 1 的结点的数目为 2，请求出该树中的叶子结点的数目。

5.3 如果已知一棵二叉树有 20 个叶子结点，有 10 个结点仅有左孩子，15 个结点仅有右孩子，求出该二叉树的结点数目。

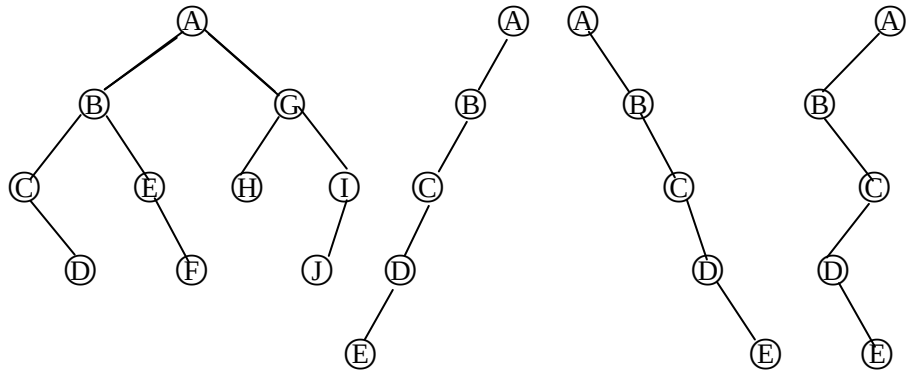
5.4 已知某完全二叉树有 100 个结点，试用三种不同的方法求出该二叉树的叶子结点数。

5.5 如果已知完全二叉树的第 6 层有 5 个叶子，试画出所有满足这一条件的完全二叉树，并指出结点数目最多的那棵完全二叉树的叶子结点数目。

5.6 在编号的完全二叉树中，判断编号为 i 和 j 的两个结点在同一层的条件是什么？

5.7 设计算法以求解编号为 i 和 j 的两个结点的最近的公共祖先结点的编号。

5.8 分别求出下图中二叉树的三种遍历序列。



5.9 分别描述满足下面条件的二叉树特征：

（1）先序序列和中序序列相同。

（2）先序序列和后序序列相反。

5.10 证明：由二叉树的先序序列和中序序列能唯一确定一棵二叉树，并分别由下面的两个序列构造出相应的二叉树：

①先序：ABCDEFGHI　②先序：ABCDEFGHIJ

中序：ADECFBGIH　中序：BDECAGIJHF

5.11 证明：由二叉树的后序序列和中序序列能唯一确定一棵二叉树，并分别由下面的两个序列构造出相应的二叉树：

①后序：DCFEBIHGA　②后序：DECBGIHFA

中序：DCBFEAGHI　中序：DCEBAFHGI

5.12 已知一棵二叉树的先序、中序和后序序列如下，其中各有一部分未给出其值，请构造出该二叉树。

先序：A_CDEF_H_J

中序：C_EDA_GFI_

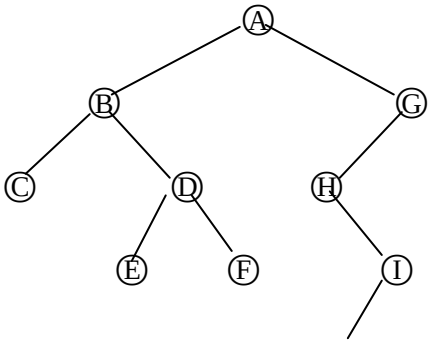
后序：C_ _BHGJI_ _

5.12 证明：任意一棵非空的二叉树的先序序列的最后一个结点一定是叶子结点。

5.13 用反例证明：由二叉树的先序序列和后序序列不能唯一确定一棵二叉树。

5.14 设计算法以输出二叉树中先序序列的前 k（k>0）个结点的值。

5. 15 设计算法按中序次序依次输出各结点的值及其对应的序号。例如，下图中的二叉树的输出结果是(C, 1) (B, 2) (E, 3) (D, 4) (F, 5) (A, 6) (H, 7) (J, 8) (I, 9) (G, 10)。



Ⓙ

5.16 设计算法以输出每个结点到根结点之间的路径上的所有结点的值。

5.17 设计算法将一棵以二叉链表形式存储的二叉树转换为顺序存储形式存储到数组A[n]中，并将其中没有存放结点值的数组元素设置为NULL。

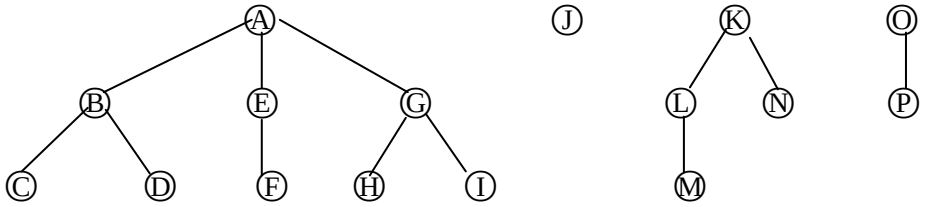
5.18 设计算法将一棵以顺序存储方式存储在数组A中的二叉树转换为二叉链表形式。

5.19 分别设计出先序、中序和后序遍历二叉树的非递归算法。

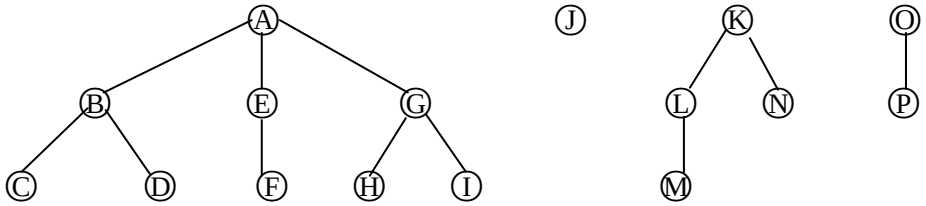
5.20 设计算法将值为x的结点作为右子树的（后序序列的）第一个结点的左孩子插入到后序线索二叉树中。

5.21 分别设计出先序、中序和后序线索化算法。

5.22 分别画出下图所示的森林的双亲表示形式、孩子链表表示形式和二叉链表表示形式。



5.23 将下图中的森林转换为对应的二叉树。

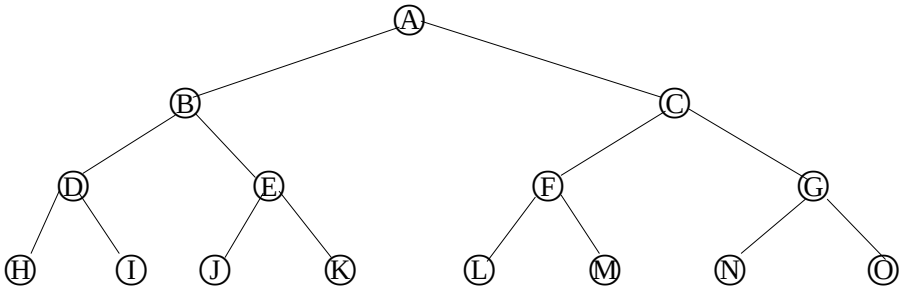


5.24 设计算法以实现将以二叉链表形式存储的树（森林）转换为对应的双亲表示形式。

5.25 已知树（森林）的高度为4，所对应的二叉树的先序序列为ABCDE，请构造出所有满足这一条件的树或森林。

5.26*设计算法将一个以孩子链表形式表示的森林转换为二叉链表形式。

5.27 将下图中的二叉树转换为对应的森林。



5.28 设计算法按先序次序输出森林中每个结点的值及其对应的层次数。

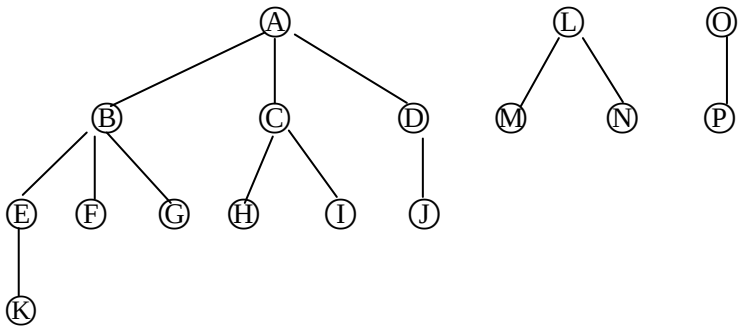
5.29 设计算法以求解森林的高度。

5.30 设计算法以输出森林中的所有叶子结点的值。

5.31 设计算法逐层输出森林中的所有结点的值。

5.32*设计算法将森林中的结点以广义表的形式输出。例如，下图中的森林的输出结果为：

(A,((B,((E,(K)),F,G)),(C,(H,I)),(D,(J)))) , (L,(M,N)), (O,(P))



5.33 以数据集合{4,6,8,10,12,15,18,20,22}中的元素为叶子结点的权值构造一棵哈夫曼树，并计算其带权路径长度。

5.34 已知一个文件中仅有 10 个不同的字符，各字符出现的个数分别为100,150,180,200,260,300,350,390,400,500。试对这些符号重新编码，以压缩文件的规模，并求出其压缩后的规模以及压缩比（压缩前后的规模比）。

5.35 设计一个程序以对文件进行压缩，计算其压缩比例，然后对所压缩的文件进行还原。

5.36 设计算法以产生哈夫曼树中各叶子结点的哈夫曼编码。

第 6 章习题

6.1 有 n 个选手参加的单循环比赛要进行多少场比赛？试用图结构描述。若是主客场制的联赛，又要进行多少场比赛？

6.2 证明下列命题：

（1）在任意一个有向图中，所有顶点的入度之和与出度之和相等。

（2）任一无向图中各顶点的度的和一定为偶数。

6.3 一个强连通图中各顶点的度有什么特点？

6.4 证明：有向树中仅有 n-1 条弧。

6.5 证明树的三个不同定义之间的等价性。

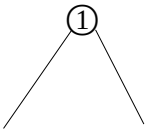
6.6 已知有向图 G 用邻接矩阵存储，设计算法以分别求解顶点 vi 的入度、出度和度。

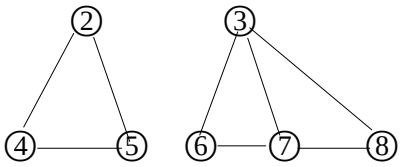
6.7 已知图 G 用邻接矩阵存储，设计算法以分别实现函数 firstadj(G，v)和 nextadj(G,v,w)。

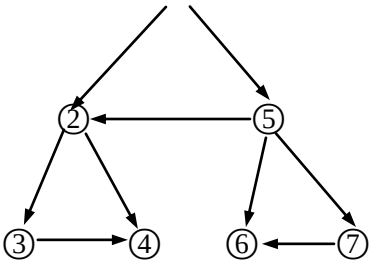
6.8 设图 G 用邻接矩阵 A[n+1,n+1]表示，设计算法以判断 G 是否是无向图。

6.9 已知图 G 用邻接表存储，设计算法输出其所有边或弧。（假设各表头指针在数组A[n+1]中）

6.10 对下列图，分别执行 dfs(1)和 dfs(5)，写出遍历序列，并构造出相应的 dfs 生成树。







6.11 对例 5－3 中的图 G（图 5－13 所示）的邻接表，不用还原出原图，请执行 dfs(1)，写出遍历序列，并构造出相应的 dfs 生成树。

6.12 设计算法以判断顶点 v_i 到 v_j 之间是否存在路径？若存在，则返回 TRUE，否则返回 FALSE。

6.13 **设计算法**以判断无向图 G 是否是连通的，若连通，返回 TRUE，否则返回 FALSE；

6.14 设 G 是无向图，设计算法求出 G 中的边数。（假设图 G 分别采用邻接矩阵、邻接表以及不考虑具体存储形式，而是通过调用前面所述函数来求邻接点）

6.15 设 G 是无向图，设计算法以判断 G 是否是一棵树，若是树，则返回 TRUE，否则返回 FALSE；

6.16 设 G 是有向图，设计算法以判断 G 是否是一棵以 v_0 为根的有向树，若是返回 TRUE，否则返回 FALSE；

6.17 在图 G 分别采用邻接矩阵和邻接表存储时，分析深度遍历算法的时间复杂度。

6.18 设连通图用邻接表 A 表示，设计算法以产生 dfs（1）的 dfs 生成树，并存储到邻接矩阵 B 中。

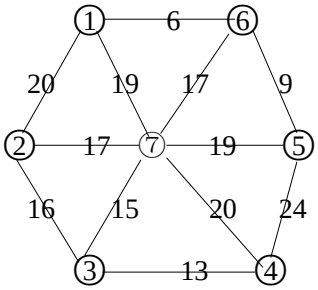
6.19 在图 G 分别采用邻接矩阵和邻接表存储时，分析广度遍历算法的时间复杂度。

6.20 设计算法以求解从 v_i 到 v_j 之间的最短路径。（每条边的长度为 1）

6.21 设计算法以求解距离 v_0 最远的一个顶点。

6.22 设计算法以求解二叉树 T 中层次最小的一个叶子结点的值。

6.23 分别用 prim 算法和 Kruskal 算法求解下图的最小生成树，标注出中间求解过程的各状态。



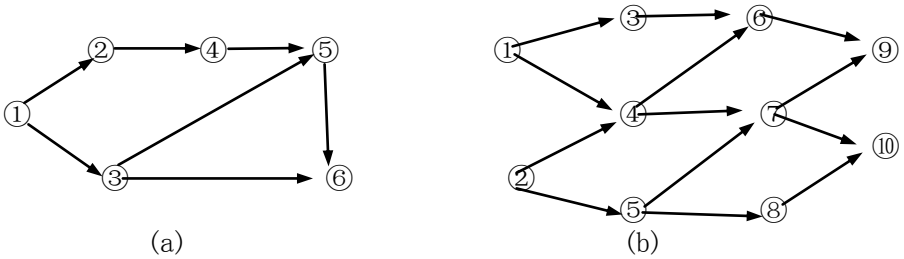
6.24 在图分别采用邻接矩阵和邻接表存储时，prim 算法的时间复杂度是否一致？为什么？

6.25 在实现 Kruskal 算法时，如何判断某边和已选边是否构成回路？

6.26 对下列 AOV 网，完成如下操作：

（1）按拓扑排序方法进行拓扑排序，写出中间各步的入度数组和栈的状态值，并写出拓扑序列。

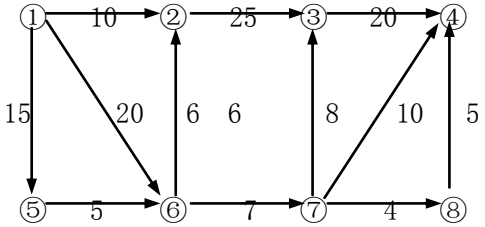
（2）写出左图所示 AOV 网的所有的拓扑序列。



（1）题图

6.27 分析在图分别采用邻接矩阵和邻接表存储时的拓扑排序算法的时间复杂度。

6.28 对下面的图，求出从顶点 1 到其余各顶点的最短路径。



6.29 分析在图分别采用邻接矩阵和邻接表存储时，求最短路径的 Dijkstra 算法的时间复杂度。

第 7 章习题

7.1 若简单顺序查找算法所要查找的元素的下标从 0 开始，因而不能用监视哨，故查找失败时要返回-1。试设计相应的算法。

7.2 对有序数据表（5,7,9,12,15,18,20,22,25,30,100），按二分查找方法模拟查找元素 10 和 28，并分别画出其搜索过程。

7.3 构造有 20 个元素的二分查找的判定树，并求解下列问题：

（1）各元素的查找长度最大是多少？

（2）查找长度为 1、2、3、4、5 的元素各有多少？具体是哪些元素？（假设下标从 0 开始）

（3）查找第 13 个元素依次要比较哪些元素？

7.4 对有 n 个元素的有序表按二分查找方法查找时，最大的查找长度是多少？

7.5 设计算法以构造有 n 个元素（下标范围从 1 到 n）的二分查找判定树。

7.6 判断题：若二叉树中每个结点的值均大于其左孩子的值，小于其右孩子的值，就一定是二叉排序树。

7.7 设计算法，求出给定二叉排序树中值为最大的结点。

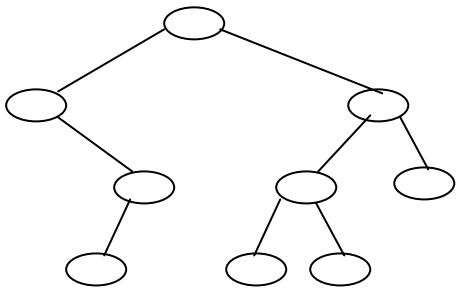
7.8 设计算法，对给定的二叉排序树，求出在等概论情况下的平均查找长度。

7.9 对给定的二叉树，假设其中各结点的值均不相同，设计算法以判断该二叉树是否是二叉排序树。

7.10 对给定的数组数据，其不同的输入序列是否一定可以构造出不同的二叉排序树？

7.11 以数据集合{1,2,3,4,5,6}的不同序列为输入构造 5 棵高度为 6 的二叉排序树。

7.12 已知一棵二叉排序树如下，其各结点的值虽然未知，但其中序序列为1,2,3,4,5,6,7,8,9。请标注各结点的值。



7.13 已知散列表地址区间为0～9,散列函数为H(k)=k % 7,采用线性探测法处理冲突。将关键字序列11,22,35,48,53,62,71,85依次存储到散列表中，试构造出该散列表，并求出在等概论情况下的平均查找长度。

7.14 设散列函数为H(k)=k % 7，采用拉链法处理冲突，将关键字序列10,26,38,43,55,69,72,88,100,92依次存储到散列表中，并求出在等概论情况下的平均查找长度。

7.16 设关键字序列为JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC,散列函数为H(k)=序号/4，其中序号指首字母在字母表中的序号，例如，字母A的序号为1。采用拉链法处理冲突，构造出该散列表，并求出在等概论情况下的平均查找长度。

7.17 已知散列表的地址区间为0～10，散列函数为H(k)=k % 11，采用线性探测法处理冲突。设计算法在其中查找值为x的元素，若查找成功，返回其下标，否则返回－1。

7.18 已知散列表地址区间为0～10，散列函数为H(k)=k % 11，采用线性探测法处理冲突。设计算法将值为x的元素插入到表中。

7.19 假设散列函数为H(k)=k % 7，采用拉链法处理冲突。设计算法在其中查找值为x的元素。若查找成功，返回其所在结点的指针，否则返回NULL。

7.20*已知散列表地址区间为0～10，散列函数为H(k)=k % 11，采用线性探测法处理冲突。设计算法删除其中值为x的元素。

第8章习题

8.1 直接插入排序算法分别在什么情况下可以达到最好和最坏的情况？分别要比较和移动多少次？相应的时间复杂度分别是多少？

8.2 直接插入排序能否保证在每一趟都能至少将一个元素放在其最终的位置上？是否是稳定的排序算法？

8.3 对下面数据表，写出采用希尔排序算法排序的每趟的结果。

(78 100 120 25 85 40 90 15 60 35 105 50 30 10 28 12)

8.4 对下面数据表，写出采用冒泡排序算法排序的每趟的结果，并标明数据移动情况。

(105 50 30 25 85 40 100 12 10 28)

8.5 对下面数据表，写出采用快速排序算法排序的每趟的结果，并标明每趟的数据移动情况。

(50 30 120 25 85 40 100 12 90 15 60 35 105 78 10 28)

8.6 已知数组 A[n]中的元素为整型，设计算法将其中所有的奇数调整到数组的左边，而将所有的偶数调整到数组的右边，并要求时间复杂度为 O(n)。

8.7 已知数组 A[n]中的元素为整型，设计算法将其调整为三部分，其中左边所有元素为 3 的倍数，中间所有元素除 3 余 1，右边所有元素除 3 余 2，并要求时间复杂度为 O(n)。

8.8 设计算法以实现如下功能：不用完整排序，求解出按大小关系为第 k 位的元素。

8.9 判断下列各序列是否是堆：

(a) (100,60,80,90,40,20,30,70,35)

(b) (100,80,90,60,40,20,70,30,35)

(c) (100,90,80,40,20,70,30,60,35)

(d) (20,30,40,35,60,80,90,100,70)

8.10 将下面数据表分别调整为大根堆和小根堆。

(50 30 120 25 85 40 100 12 90 15 60 35 105 78 10 28)

8.11 由初始建堆过程的讨论可知，调整过程可以是递归形式的，请写出这一递归形式的算法。

8.12 对长度为 10000 的数据表，如果在不用完整排序的情况下，要求找出其中最大的 10 个数，应选用何种排序算法最节省时间？

8.13 如果递增有序的数据表 A、B 分别代表一个集合，设计算法以求解此类集合的交、并差等运算